Addressing

On the Naming and Binding of Network Destinations

Jerome H. Saltzer

M.I.T. Laboratory for Computer Science,
545 Technology Square,
Cambridge, MA 02139,
U.S.A.

This brief paper offers a perspective on the subject of names of destinations in data communication networks. It suggests two ideas: First, it is helpful to distinguish among four different kinds of objects that may be names as the destination of a packet in a network. Second, the operating system concept of binding is a useful way to describe the relations among the four kinds of objects. To illustrate the usefulness of this approach, the paper interprets some more subtle and confusing properties of two real-world network systems for naming destinations.

Published in: "Local Computer Networks", North-Holland Publishing Company.

This research was suuported in part by the Defense Advanced Research Projects Agency of the United States Government and monitored by the Office of Naval Research under contract number N00014-75-C-0661.

What is the Problem?

Despite a very helpful effort of John Shoch [1] to impose some organization on the discussion of names, addresses, and routes to destinations in computer networks, these discussions continue to be more confusing than one would expect. This confusion stems sometimes from making too tight an association between various types of network objects and the most common form for their names. It also stems from trying to discuss the issues with too few well-defined concepts at hand. This paper tries a different approach to develop insight, by applying a perspective that has proven helpful in the corresponding area of computer operating systems.

Operating systems have a similar potential for confusion concerning names and addresses, sicne there are file names, unique identifiers, virtual and real memory addresses, page numbers, block numbers, I/O channel addresses, disk track addreses, a seemingly endless list. But most of that potential has long been rendered harmless by recognizing that the concept of binding provides a systematic way to think about naming [2]. (Shoch pointed out this opportunity to exploit the operating system concept; in this paper we make it the central theme.) In operating systems, it was apparent very early that there were too many different kinds of identifiers and therefore one does not get much insight by trying to make a distinction just between names and addresses. It is more profitable instead to look upon all identifiers as examples of a single phenomenon, and ask instead "where is the context in which a binding for this name (or address, or indentifier, or whatever) will be found?", and "to what object, identified by what kind of name, is it therein bound?" This same approach is equally workable in data communications networks.

To start with, let us review Shoch's suggested terminology in its broadest form:

- a name identifies what you want,
- an adddress identifies where it is, and
- an route identifies a way to get there.

There will be no need to tamper with these definitions,, but it will be seen that they will leave a lot of room for interpretation. Shoch's suggestion implies that there are three abstract concepts that together provide an intellectual for discussion. In this paper, we propose that a more mechanical view may lead to an easier-to-think-with set of concepts. This more mechanical view starts by listing the kinds of things one finds in a communication network.

Types of Network Destinations, and Bindings among them

In a data communication network, when thinking about how to describe the destination of a packet, there are several types of things for which there are more than one instance, so one attaches names to them to distinguish one instance from another. Of these several types, four turn up quite often:

1. Service and Users. These are the functions that one uses, and the clients that use them. Examples of services are one that tells the time of day, one that performs acounting, or one that forwards packets. An example of a client is a particular desktop computer.

2. Nodes. These are computers that can run services or user programs. Some nodes are clients of the network, while others help implement the network by running forwarding services. (We will not need to distinguish between these two kinds of nodes.)

3. Network attachment points. These are the ports of a network, the places where a node is attached. In many discussions about data communication networks, the term "address" is an identifier of a network attachment point.

4. Paths. These run between network attachment points, traversing forwarding nodes and communication links.

We might note that our first step, the listing and characterization of the objects of discussion, is borrowed from the world of abstract data types. Our second step is to make two observations about the naming of network objects, the first about form and the second about bindings.

First, one is free to choose any form of name that seems helpful -- binary identifiers, printable character strings, or whatever, and they may be chosen from either a flat or hierarchical name space. There may be more than one form of name for a single type of object. A node might, for example, have both a hierarchical character string name and a unique binary identifier.

There are two semantic traps that one can fall into related to name form. First, the word "name" is, in the network world, usually associated with a printable character string, while the word "address" is usually associated with machine- interpretable binary strings. In the world of systems and languages, the term "print name" is commonly used for the first and "machine name" or "address" for the second, while "name" broadly encompasses both forms. (In this paper we are using the broad meaning of "name".) The second semantic trap is to associate some conventional form of name for a particular type of object as a property of that type. For example, services might be named by character strings, nodes by unique ID's, and network attachment points named by hierarchical addresses. When one participant in a discussion assumes a particular name form is invariably associated with a particular type of object and another doesn't, the resulting conversation can be very puzzling to all participants.

The second observation about the four types of network objects listed above is that most of the naming requirements in a network can simply and concisely be described in terms of bindings and changes of bindings among the four types of objects. To wit:

1. A given service may run a one or more nodes, and may need to move from one node to another without losing its identity as a service.

2. A given node may be connected to one or more network attachment points, and may need to move from one attachment point to another without losing its identity as a node.

3. A given pair of attachment points may be connected by one or more paths, and those paths may need to change with time without affecting the identity of the attachment points.

(This summary of network naming requirements is intentionally brief. An excellent in-depth review of these requirements can be found in a recent paper by Sunshine [3].)

Each of these three requirements includes the idea of preserving identity, whether of service, node or attachment point. To preserve an identity, one must arrange that the name used for identification not change during moves of the kind required. Is the associations amove services, nodes, attachment points and routes are maintained as lists of bindings this goal can easily be met. Whether or not all the flexibility implied by these possibilities should be provided in a particular network design is a matter of engineering judgement. A judgement that a particular binding can be made at network design time and will never need to be changed (e.g. a particular service might always run at a particular node) should not be allowed to confuse the question of what names and bindings are in principle present. In principle, to send a data packet to a service one must discover three bindings:

1. find a node on which the required service operates,

2. find a network attachment point to which that node is connected,

3. find a path from this attachment point to that attachment point.

There are, in turn, three conceptually distinct binding services that the network needs to provide:

1. Service name resolution, to indentify the nodes that run the service.

2. Node name location, to identify attachment points that reach the nodes found in 1.

3. Route service, to identify the paths that lead from the requestor's attachment point to the ones found in 2.

At each level of binding, there can be several alternatives, so a choice of which node, which attachment point, and which path must be made. These choices are distinct, but can interact. For example, one might chose the node only after first looking over the various paths leading to the possible choices. In this case, the network tables may only produce a partial binding, which means that an enquiry produces a list of answers rather than a single one. The final binding choice may be delayed until the last moment and recorded outside the three binding services provided within the network.

There is a very important subtlety about bindings that often leads designers astray. Suppose we have recorded in a network table the fact that the "Lockheed DIALOG Service" is running on node "5". There are actually three different bindings involved here but only one of these three is recorded in this table and changeable by simply adjusting the table.

1. The name "Lockheed DIALOG Service" is properly associate with a specific service, management, and collection of stored files. One does not usually reassign such a name to a different service. The association of the name with the service is quite permanent, and because of that permanence is not usually expressed in a single, easily changed table.

2. Similarly, the name "5" is assigned to a particular node on a fairly long-term basis, without the expectation that it will change. So that assignment is also not typically expressed in a single, easily changed table.

3. The fact that "DIALOG" is now operating on node "5"is the one binding that our table does express, because we anticipate that this association might reasonably change. The function of our table is to allow us to express changes such as "DIALOG" is now operating at node "6" or the "Pipe-fitting Service" is now operating at node "5".

The design mistake is to believe that this table allows one to give the Lockheed DIALOG service a new name, merely by changing this table entry. That is not the function of this table of bindings, and such a name change is actually quite difficult to accomplish, since the association in question is not usually expressed as a binding in a single table. One would have to change not only this table, but also user programs, documentation, scribbled notes and advertising copy to accomplish such a name change.

Some Real-World Examples

Although the ideas outlined so far seem fairly straightforward, it is surprisingly easy to find real-world examples that pose a challenge in interpretation. In the Xerox/DEC/Intel Ethernet [5, 6], for example, the concept of a network attachment point is elusive, because it collapses into the node name. A node can physical attach to an Ethernet anywhere along it; the node brings with it a 48-bit unique identifier that its interfaces watches for in packets passing by. This identifier should probably be thought of as the name of a network attachment point, even though the physical point of attachment can be anywhere. At the same time, one can adopt a policy that the node will supply from its own memory the 48-bit identifier that is to be used by the Ethernet interface, so a second, equally reasonable, view (likely to be taken elsewhere in the network in interpreting the meaning of these identifiers) is that this 48-bit identifier is the name of the node itself. From a binding perspective this way of using the Ethernet binds the node name and the network attachment point name to be the same 48-bit unique identifier.

This permanent binding of node name to attachment point name has several network management advantages:

- a node can be moved from one physical location to another without changing any network records.

- one level of binding tables is omitted. This advantage is particularly noticeable in implementing internetwork routing.

- a node that is attached to two Ethernets can present the same attachment point name to both networks, which simplifies communication among internet routers and alternate path finding.

But permanent binding also produces a curiosity if is happens that one wants one node to connect to two attachment points on the same Ethernet. The curiosity arises because the only way to make the second attachment point independently addressableby others is to allow the node to use two different 48-bit identifiers, which means that some other network records (the ones that interpret the ID to be a node name) will likely be fooled into believing that there are not one, but two nodes. To avoid this confusion, the same 48-bit identifier could be used in both attachment points, but then there will be no way intentionally to direct a message to one rather than the other. One way or another, the permanent binding of attachment point name to node name has made some function harder to accomplish, though the overall effect of the advantages probably outweighs the lost function in this case.

For another example, the ARPANET NCP protocol provides character string names that appear, from their mnemonics, to be node names or service names, but in fact they are the names of network attachment points [6]. Thus the character string name RADC-Multics is the name of the network arrachment point at ARPANET IMP 18, port 0, so reattaching the node (a Honeywell 68/80 computer) to another network attachment point requires either that the users learn a new anme for the service or else a change of tables in all other nodes. Changing tables superficially appears to be what rebinding is all about, but the need to change more than one table is the tip-off that something deeper is going on. What is actually happening is the change of the permanent name of the network attachment point. We can see this more clearly by noting that a parallel attachment of that Honeywell 68/80 to a second ARPANET port would be achievable only by assigning a second character string identity; this requirement emphasizes that the name is really of the attachment point, not the node. Unfortunately, because of their mnemonic value, the ARPANET NCP name mnemonics are often thought of as service names. Thus one expects that that the Rome Air Development Center Multics service is operated on the node reached by the name RADC-Multics. This particular assumption doesn't produce any surprises. But any one of the four Digital PDP-10 computers at Bolt, Beranek and Newman can accept mail for any of the others, as can the groups of PDP-10's at the USC Information Sciences Institute, and at the Massachusetts Institute of Technology. If the node to which ones tries to send mail is down, the customer must realize that the same service is available by asking for a different node, using what appears to be a different service name. The need for a customer to realize that he must give a different name to get the same service comes about because in the ARPANET the name is not of a service that is bound to a node that is bound to an attachment point, but rather it is directly the name of an attachment point.

Finally, confusion can arise because the three conceptually distinct binding services (service name resolution, node name location, and route dispensing) may not be mechanically distinct. There is usually suggeset only one identifiable service, a "name server". The name server starts with a service name and returns a list of network attachment points that can provide that service. It thereby performs both the first and second conceptual binding services, though it may leave to the customer the final choice of which attachment point to use. Path choice may be accomplished by a distributed routing algorithm that provides the final binding service without anyone noticing it.

With this model of binding among services, nodes, network attachment points, and paths in mind, one possible interpretatio of Shoch's "names", "addresses" and "routes" is as follows:

1. Any of the four kinds of objects (service, node, network attachment point, and path) may have a name, though Shoch would restrict that term to human-readable character strings.

2. The address of an object is a name (in the broad sense, not Shoch's restricted sense) of the object it is bound to. Thus, an address of a service is the name of some node that runs it. An address of a node is the name of some network attachment point to which it connects. An address of a network attachment point (a concept not usually discussed) can be taken to be the name of a path that leads to it. This interpretation captures Shoch's meaning "An address indicates where it is," but does not very well match Shoch's other notion that an address is a machine-processable, rather than a human-processable form of identification. This is probably the primary point wher our perspectives differ on which definitions provide the most clarity.

3. A route is a more sophisticated concept. A route to either a network attachment point or a node is just a path, as we have been using the term. Because ~~a~~ single node can run several services at once, a route to a service consists of a pth to the network attachment point of a node that runs the service, plus some identification of which activity within that node runs the service (e.g., a "socket identifier" in the PUP internet [4] or the ARPA Internet [7] protocols). But note that a route may actually consist of a series of names, typically a list of forwarding name nodes or attachment points and the names used by the forwarding nodes for the paths between them.

Whether or not one likes this particular interpretation of Shoch's terms, it seems clear that there are more than three concepts involved, so more than three labels are needed to discuss them.

Summary

This paper has argued that some insight into the naming of destinations in a network can be obtained by recognizing four kinds of named objects at or leading to every destination (services, nodes, attachment points, and routes) and then identifying three successive, changeable, bindings (service to node, node to attachment point, and attachment point to route). This perspective, modeled on analogous successive bindings of storage management systems (file--storage region--physical location) and virtual memories (object--segment--page--memory block) provides a systematic explanation for some design problems that are encountered in network naming systems.

Acknowledgements

Discussions with Dave D. Clark, J. Noel Chiappa, David P. Reed, and Danny Cohen helped clarify the reasoning used here. John F. Shoch provided both inspiration and detailed comments, but should not be held responsible for the result.

References

1. Shoch, John F., "Inter-Network Naming, Addressing, and Routing," IEEE Proc. COMPCON Fall 1978, pp. 72-79. Also in Thurber, K. (ed.), Tutorial: Distributed Processor Communication Architecture, IEEE Publ. #EHO 152-9, 1979, pp. 280-287.

2. Saltzer, J. H., "Naming and Binding of Objects," in: Operating Systems, Lecture notes in Computer Science, Vol. 60, Edited by R. Bayer, New York; Springer-Verlag, 1978.

3. Sunshine, Carl A., "Addressing Problems in Multi-Network Systems," to appear in Proc. IEEE INFOCOM 82, Las Vegas, Nevada, March 30 - April 1, 1982.

4. Boggs, D. R., Shoch, J. F., Taft, E. A., and Metcalfe, R. M., "PUP: An Internetwork Architecture," IEEE Trans. on Comm. 28, 4 (April, 1980) pp. 612-623.

5. (Anonymous), "The Ethernet, A Local Area Network: Data Link Layer and Physical Layer Specifications, Version 1.0," published by Xerox Corp., Palo Alto, Calif., Intel Corp., Sunnyvale, Calif., and Digital Equipment Corp., Tewksbury, Mass., September 30, 1980.

6. Dalal, Y. K., and Printis, R. S., "48-bit Absolute Internet and Ethernet Host Numbers," Proc. Seventh Data Communications Symposium, Mexico City, Mexico, October 1981, pp. 240-245.

7. Feinler, E., and Postel, J., Ed., "ARPANET Protocol Handbook," SRI International, Menlo Park, Calif., January, 1978.